

Fig. 1

200

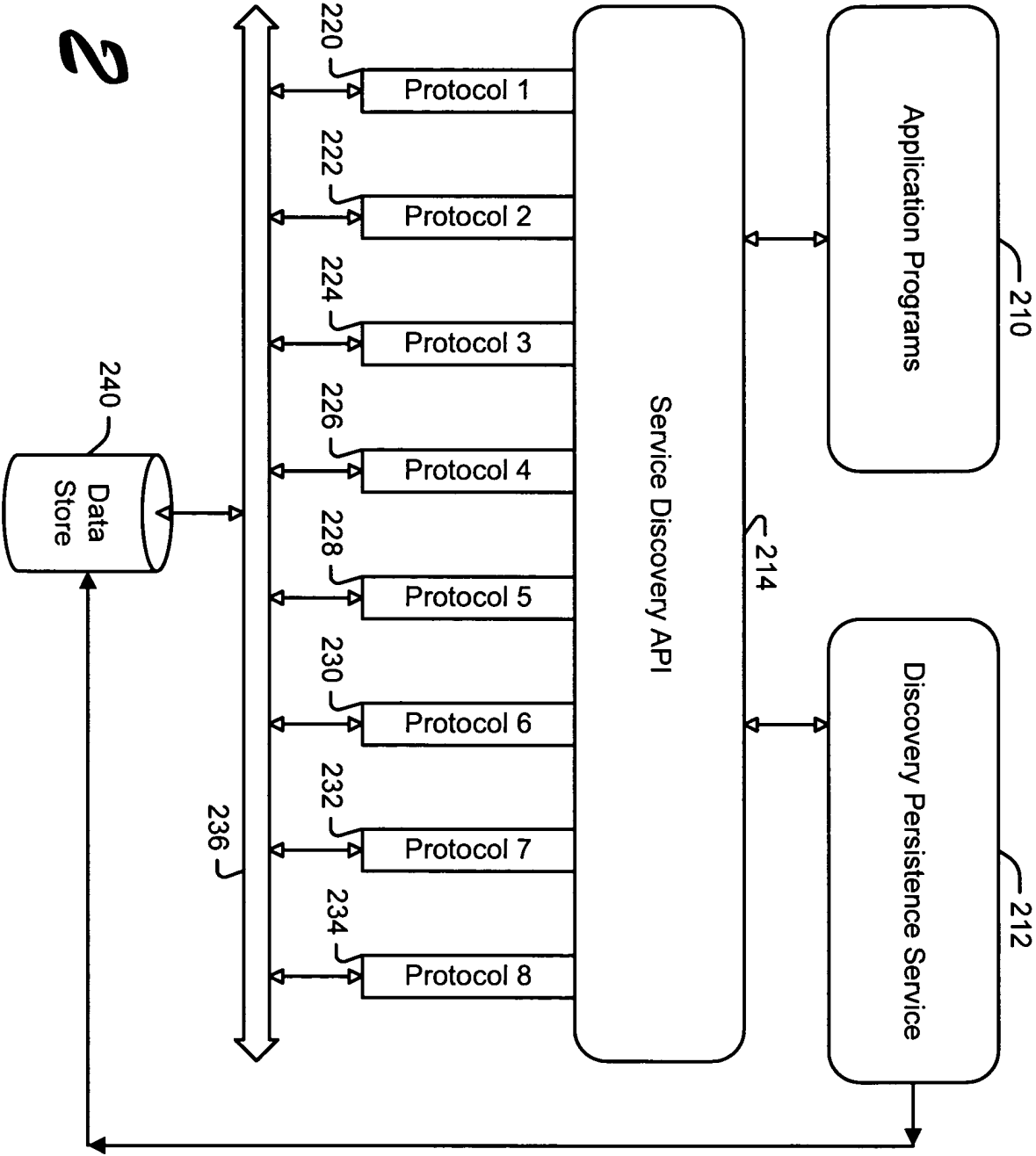


Fig. 2

300

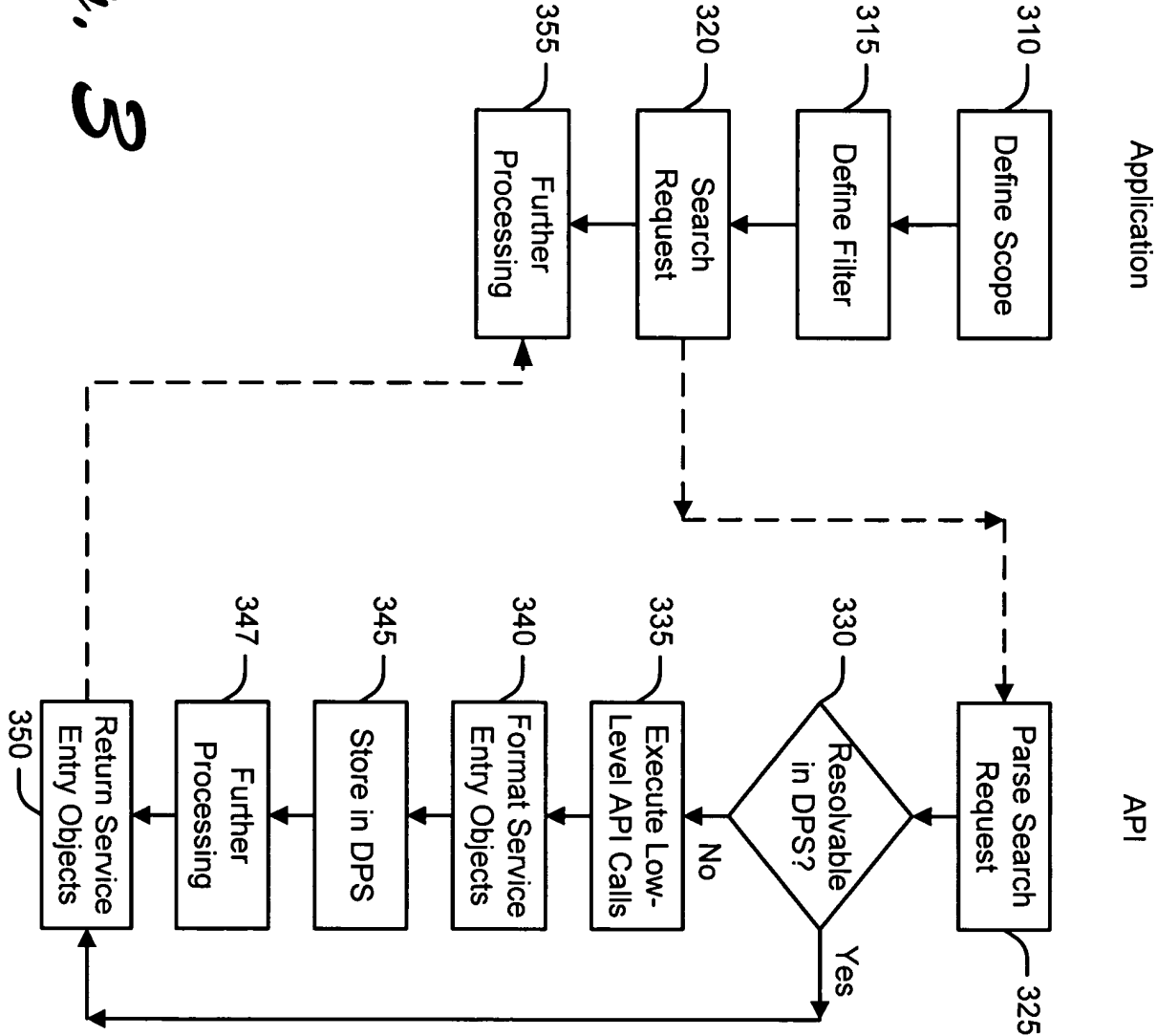


Fig. 3

400

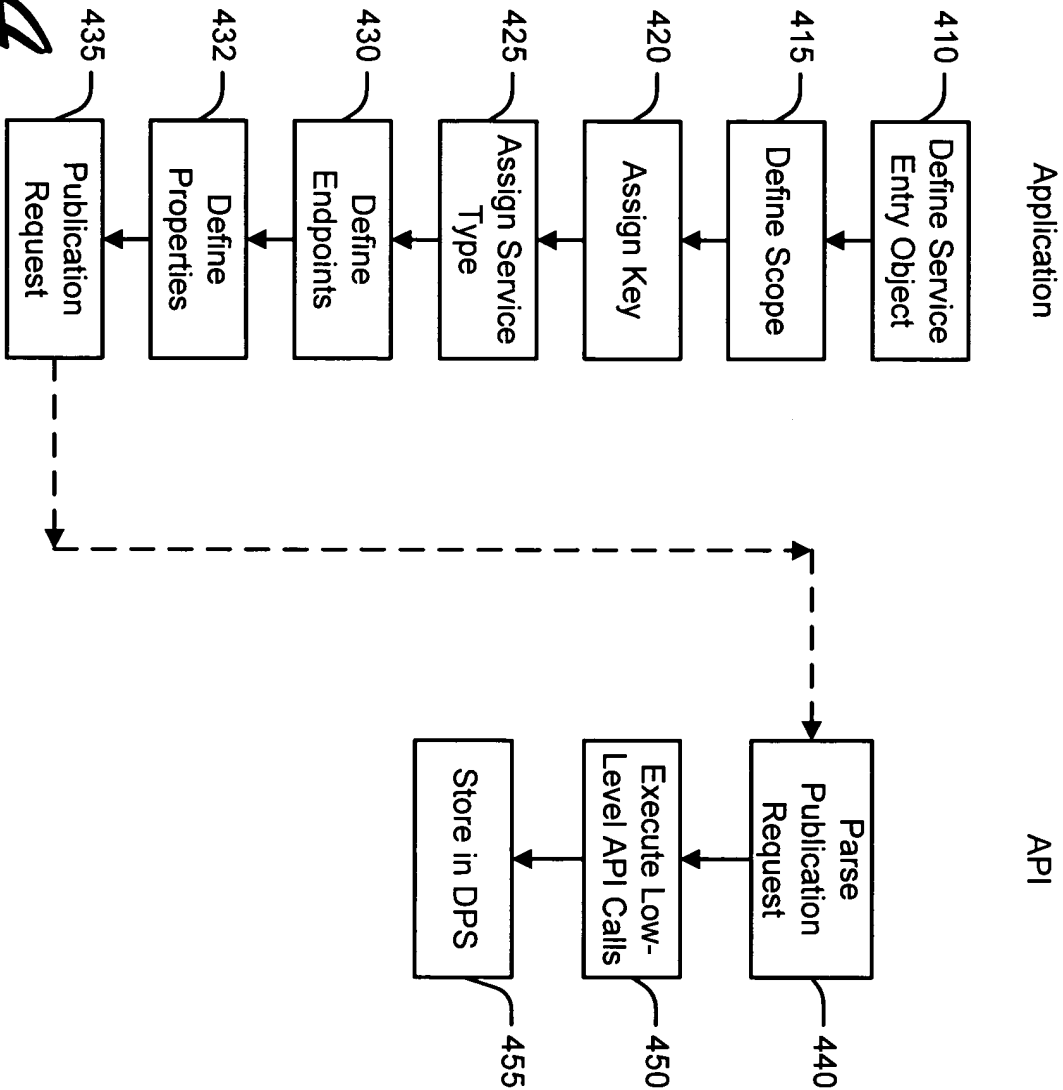


Fig. 4

500

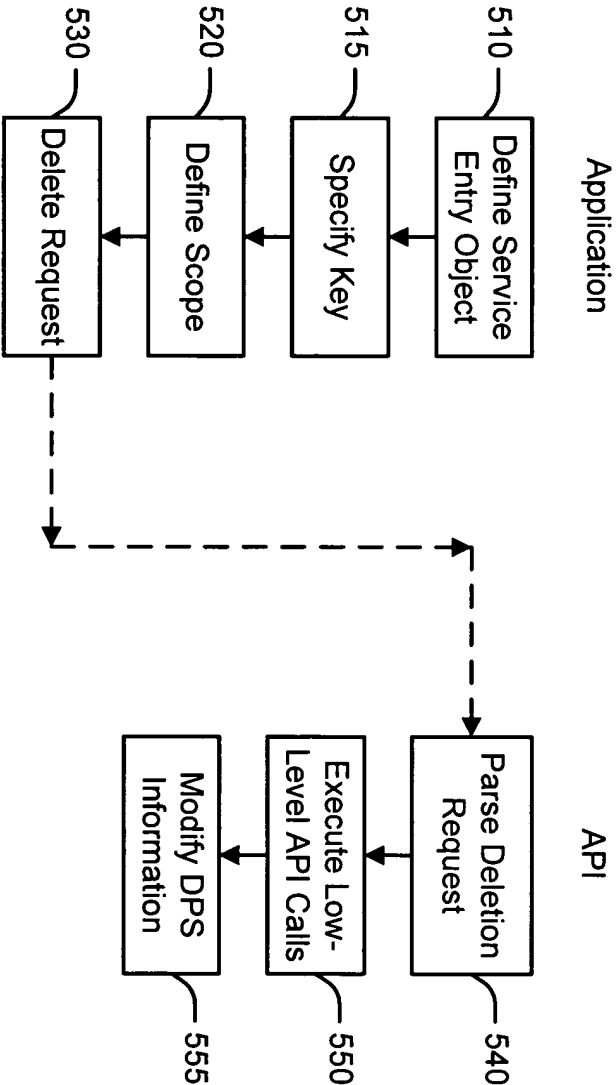


Fig. 5

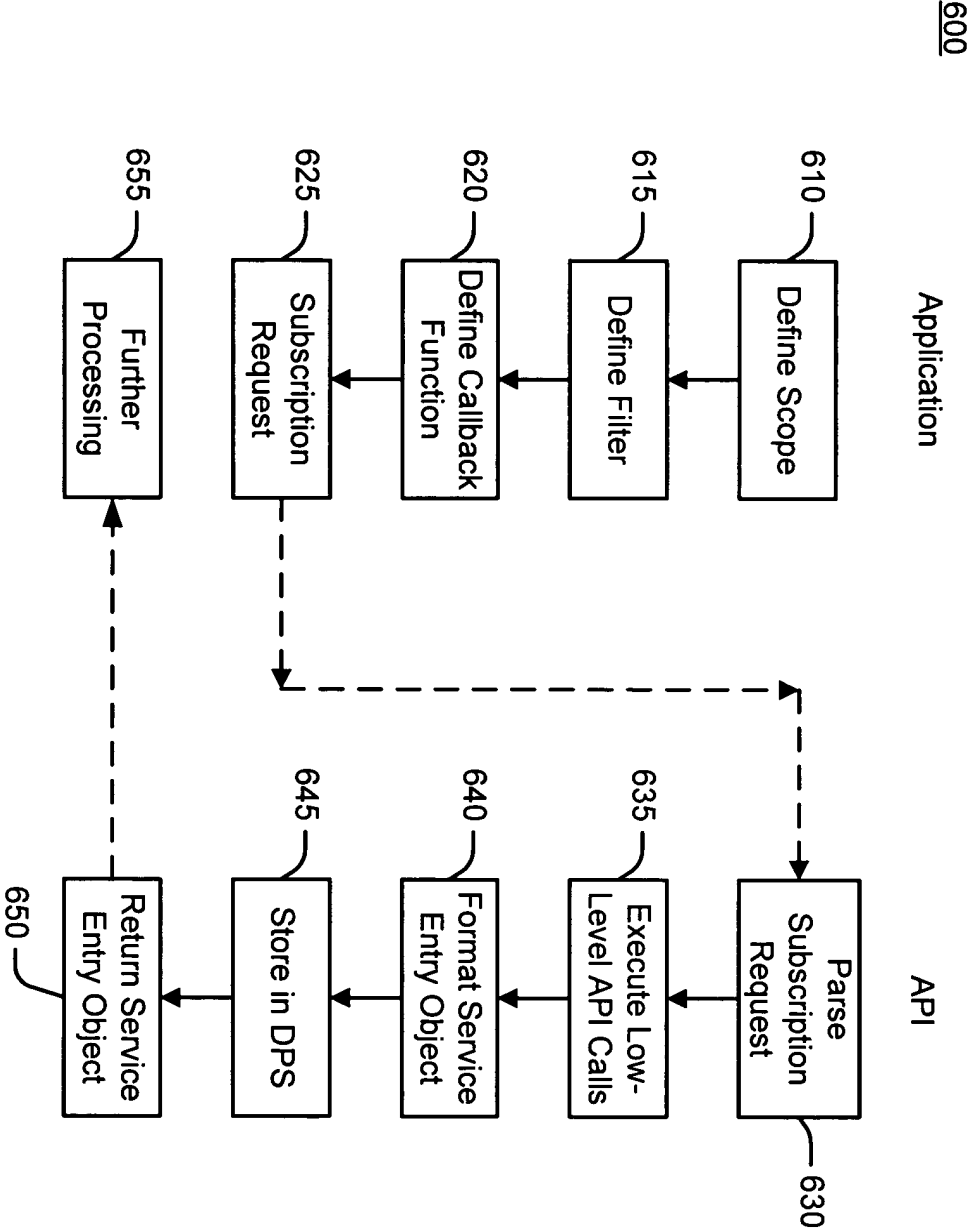


Fig. 6

200

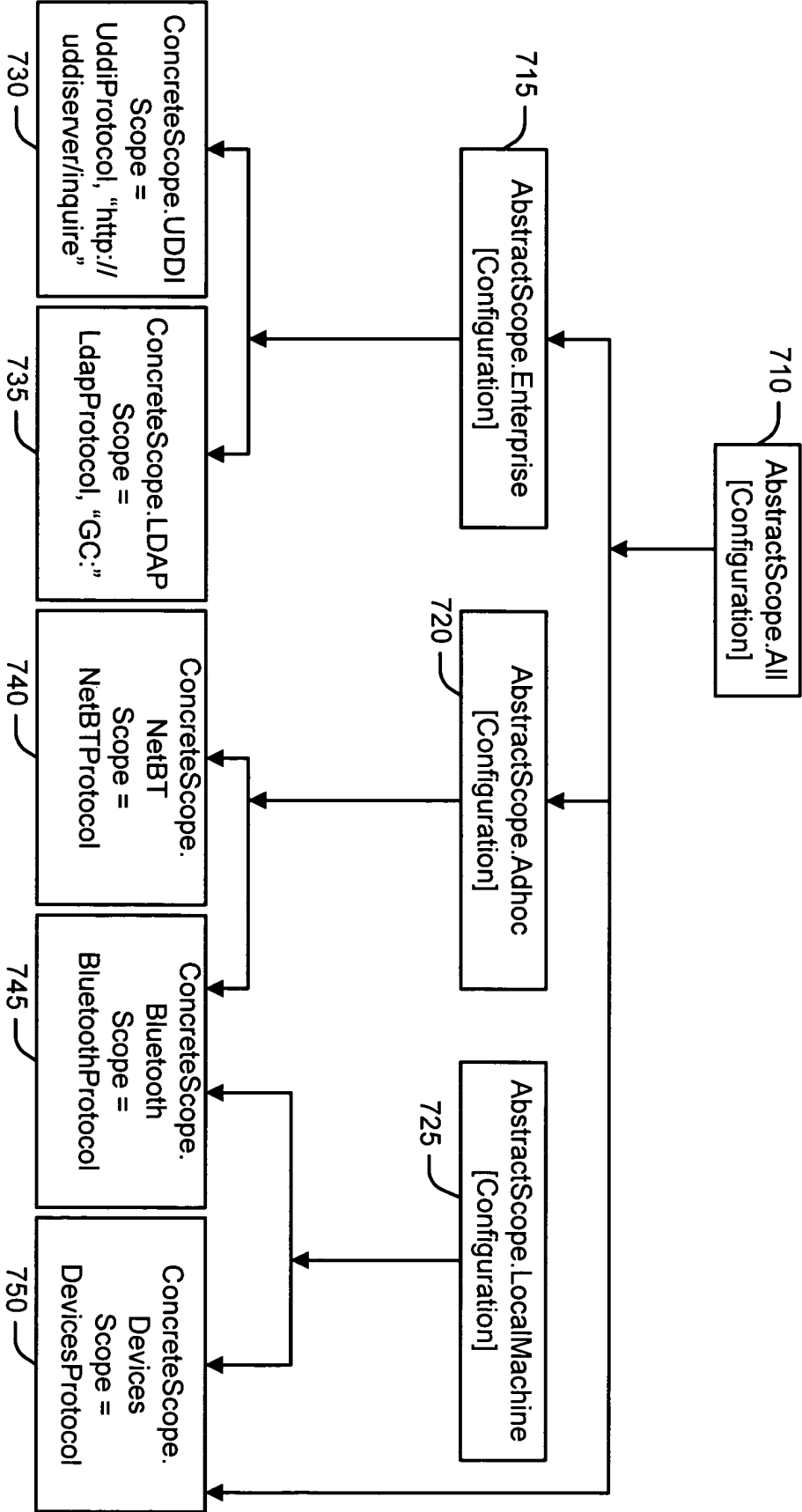


Fig. 7

```
static void SimpleFilterAD()
{
    // Declare a ServiceFinder
    System.Discovery.ServiceFinder servFind = new ServiceFinder();

    // Specify to search using Active Directory in the global catalog
    servFind.Scopes.Add (new ConcreteScope (ConcreteScope.ADProtocol, "GC:", null));

    // Filter for only printers
    SimpleFilter filter = new SimpleFilter ();
    filter.ServiceType = CommonServiceTypes.Printer;
    servFind.Filter = filter;

    // Declare the properties to match on
    filter.Properties.Add ("printcolor", "TRUE");
    filter.Properties.Add ("pagesperminute", "50");

    // Find all entities matching the filter within the scope
    ServiceEntryCollection seColl = servFind.FindAll ();

    // Report results
    Console.WriteLine ("Results Found: " + seColl.Count);

    foreach (ServiceEntry se in seColl)
    {
        Console.WriteLine ("Name: " + se.Name);
        Console.WriteLine ("Description: " + se.Description);
        If ((se.Endpoints!=null) && (se.Endpoints.Count>0))
        {
            Console.WriteLine ("Address: " + se.Endpoints[0].Address);
        }
        Console.WriteLine ("Key: " + se.Key);
        Console.WriteLine ();
    }
}
```

Fig. 8


```
static void RichFilterUDDI ()
{
    // Declare a ServiceFinder
    ServiceFinder servFind = new ServiceFinder ();

    // Specify to search using the UDDI protocol in the
    // test version of the Microsoft UDDI Business Registry
    servFind.Scopes.Add (new ConcreteScope (ConcreteScope.UddiProtocol,
        "http://test.uddi.microsoft.com/inquire", null));

    // Filter for only Web services named Fabrikam
    // that implement a specific interface i.e. the
    // tModel "uddi-org:inquiry_v2"
    RichFilter filter = new RichFilter
        ("WebService[ name = 'Fabrikam' and ServiceInterface = 'uuid:ac104dcc-d623-452f-88a7-f8acd94d9b2b' ]");

    servFind.Filter = filter;

    // Find all entities matching the filter within the scope
    ServiceEntryCollection seColl = servFind.FindAll ();

    // Report results
    Console.WriteLine ("Results Found: " + seColl.Count);
    foreach (ServiceEntry se in seColl)
    {
        Console.WriteLine ("Name: " + se.Name);
        if((se.Endpoints!=null) && (se.Endpoints.Count>0))
        {
            Console.WriteLine ("Address: " + se.Endpoints[0].Address);
        }
        Console.WriteLine ("Key: " + se.Key);
        Console.WriteLine ();
    }
}
```

Fig. 9

```
static void SimpleFilterUDDI ()
{
    // Declare a ServiceFinder.
    System.Discovery.ServiceFinder servFind = new ServiceFinder ();

    // Specify to search using the UDDI protocol in the
    // Microsoft UDDI Business Registry node.
    servFind.Scopes.Add (new ConcreteScope (ConcreteScope.UddiProtocol,
"http://uddi.microsoft.com/inquire", null));

    // Filter for only services that implement a specific interface,
    // i.e. the tModel "uddi-org:inquiry_v2".
    SimpleFilter filter = new SimpleFilter ();
    filter.ServiceInterfaces.Add(
"uuid:ac104dcc-d623-452f-88a7-f8acd94d9b2b");

    servFind.Filter = filter;

    // Find all entities matching the filter within the scope.
    ServiceEntryCollection seColl = servFind.FindAll ();

    // Report results.
    Console.WriteLine ("Results Found: " + seColl.Count);

    foreach (ServiceEntry se in seColl)
    {
        Console.WriteLine ("Name: " + se.Name);
        Console.WriteLine ("Description: " + se.Description);
        if((se.Endpoints!=null) && (se.Endpoints.Count>0))
        {
            Console.WriteLine ("Address: " + se.Endpoints[0].Address);
        }
        Console.WriteLine ("Key: " + se.Key);
    }
}
```

Fig. 10

Docket No:

MS1-1802US

11 of 19

Inventor(s):

Parham, et al.

Title:

SERVICE DISCOVERY AND PUBLICATION

Sub SimpleFilterUDDI()

' Declare a ServiceFinder.

Dim servFind = New ServiceFinder

' Specify to search using the UDDI protocol in the

' Microsoft UDDI Business Registry node.

servFind.Scopes.Add(New ConcreteScope(ConcreteScope.UddiProtocol, _
"http://uddi.microsoft.com/inquire", Nothing))

' Filter for only services that implement a specific interface,

' i.e. the tModel "uddi-org:inquiry_v2".

Dim filter As New SimpleFilter

filter.ServiceInterfaces.Add(_

"uuid:ac104dcc-d623-452f-88a7-f8acd94d9b2b")

servFind.Filter = filter

' Find all entities matching the filter within the scope.

Dim seColl As ServiceEntryCollection = servFind.FindAll()

' Report results.

Console.WriteLine("Results Found: " & seColl.Count)

Dim se As ServiceEntry

For Each se In seColl

Console.WriteLine("Name: " & se.Name)

Console.WriteLine("Description: " & se.Description)

If Not (se.Endpoints Is Nothing) AndAlso se.Endpoints.Count > 0 Then

Console.WriteLine("Address: " & se.Endpoints(0).Address)

End If

Console.WriteLine("Key: " & se.Key)

Next se

End Sub 'SimpleFilterUDDI

Fig. 11

```
static void RichFilterAD ()
{
    // Declare a ServiceFinder.
    ServiceFinder servFind = new ServiceFinder ();

    // Specify to search using Active Directory in the global catalog.
    servFind.Scopes.Add (new ConcreteScope (
        ConcreteScope.ADProtocol, "GC:", null));

    // Filter for printers where the name begins with 'Office Printer'.
    RichFilter filter = new RichFilter (
        "Printer[ name like 'Office Printer' ]");

    servFind.Filter = filter;

    // Find all entities matching the filter within the scope.
    ServiceEntryCollection seColl = servFind.FindAll ();

    // Report results
    Console.WriteLine ("Results Found: " + seColl.Count);
    foreach (ServiceEntry se in seColl)
    {
        Console.WriteLine ("Name: " + se.Name);
        if((se.Endpoints!=null) && (se.Endpoints.Count>0))
        {
            Console.WriteLine ("Address: " + se.Endpoints[0].Address);
        }
        Console.WriteLine ("Key: " + se.Key);
    }
}
```

Fig. 12

```
Sub RichFilterAD()  
    ' Declare a ServiceFinder.  
    Dim servFind As New ServiceFinder  
  
    ' Specify to search using Active Directory in the global catalog.  
    servFind.Scopes.Add(New ConcreteScope(_  
        ConcreteScope.ADProtocol, "GC:", Nothing))  
  
    ' Filter for printers where the name begins with 'Office Printer'.  
    Dim filter As New RichFilter("Printer[ name like 'Office Printer' ]")  
  
    servFind.Filter = filter  
  
    ' Find all entities matching the filter within the scope.  
    Dim seColl As ServiceEntryCollection = servFind.FindAll()  
  
    ' Report results  
    Console.WriteLine("Results Found: " & seColl.Count)  
    Dim se As ServiceEntry  
    For Each se In seColl  
        Console.WriteLine("Name: " & se.Name)  
        If Not (se.Endpoints Is Nothing) AndAlso se.Endpoints.Count > 0 Then  
            Console.WriteLine("Address: " & se.Endpoints(0).Address)  
        End If  
        Console.WriteLine("Key: " & se.Key)  
    Next se  
  
End Sub 'RichFilterAD
```

Fig. 13

```
const string sampleServiceType = "75FF48AB-E771-4d44-9E8C-E09141112417";  
static string sampleServiceKey = Guid.NewGuid().ToString();  
  
static void Save()  
{  
    // Instantiate a new ServiceEntry object.  
    ServiceEntry service = new ServiceEntry();  
  
    // Add the scope to publish this service entry into.  
    service.Scopes.Add( new ConcreteScope( ConcreteScope.SsdpProtocol ) );  
  
    // Assign a unique key, some protocols may generate one.  
    service.Key = sampleServiceKey;  
  
    // The type of the service is specified by setting  
    // the ServiceType property.  
    service.ServiceType = new ServiceInterface( sampleServiceType );  
  
    // Add an endpoint reference to the address for accessing the service.  
    Endpoint ep = new Endpoint( Environment.MachineName, null );  
    service.Endpoints.Add( ep );  
  
    // Publish the service information synchronously  
    // into the specified scope.  
    service.Save();  
}
```

Fig. 14

```
Const sampleServiceType As String = "75FF48AB-E771-4d44-9E8C-E09141112417"  
Shared sampleServiceKey As String = Guid.NewGuid().ToString()  
  
Shared Sub Save()  
    ' Instantiate a new ServiceEntry object.  
    Dim service As New ServiceEntry  
  
    ' Add the scope to publish this service entry into.  
    service.Scopes.Add(New ConcreteScope(ConcreteScope.SsdpProtocol))  
  
    ' Assign a unique key, some protocols may generate one.  
    service.Key = sampleServiceKey  
  
    ' The type of the service is specified by setting  
    ' the ServiceType property.  
    service.ServiceType = New ServiceInterface(sampleServiceType)  
  
    ' Add an endpoint reference to the address for accessing the service.  
    Dim ep As New Endpoint(Environment.MachineName, Nothing)  
    service.Endpoints.Add(ep)  
  
    ' Publish the service information synchronously  
    ' into the specified scope.  
    service.Save()  
  
End Sub 'Save
```

Fig. 15

```
static void Delete()
{
    // Instantiate a new ServiceEntry object.
    ServiceEntry service = new ServiceEntry();

    // Specify the unique key of the service to delete.
    service.Key = sampleServiceKey;

    // Add the scope to delete this service from.
    service.Scopes.Add( new ConcreteScope( ConcreteScope.SsdProtocol ) );

    // Delete the service information synchronously
    // from the specified scope.
    service.Delete();
}
```

Fig. 16


```
Sub Delete()  
    ' Instantiate a new ServiceEntry object.  
    Dim service As New ServiceEntry  
  
    ' Specify the unique key of the service to delete.  
    service.Key = sampleServiceKey  
  
    ' Add the scope to delete this service from  
    service.Scopes.Add(New ConcreteScope(ConcreteScope.SsdpProtocol))  
  
    ' Delete the service information synchronously  
    ' from the specified scope.  
    service.Delete()  
  
End Sub 'Delete
```

Fig. 17

Docket No: MS1-1802US
 Inventor(s): Parham, et al.
 Title: SERVICE DISCOVERY AND PUBLICATION

```

public sealed class SubscriptionSample
{
    // This delegate will be called each time a notification of an
    // update on the SampleService occurs.
    private static ServiceFinder.ServiceUpdateEventHandler handler =
        new ServiceFinder.ServiceUpdateEventHandler( Handler );

    private static ServiceFinder finder = new ServiceFinder();

    public static void Subscribe()
    {
        // Initialize the finder to look for 'castle' home networks.
        finder.Scopes.Add( new ConcreteScope(
            ConcreteScope.SsdProtocol ) );
        finder.Filter = new SimpleFilter(
            CommonServiceTypes.HomeNetwork );

        // Add our handler to the delegate for service updates.
        finder.ServiceUpdate += handler;
    }

    public static void Unsubscribe()
    {
        // Remove our handler from the delegate for service updates.
        finder.ServiceUpdate -= handler;
    }

    // This is the callback function that was registered with the delegate.
    private static void Handler( object sender, ServiceUpdateEventArgs e )
    {
        // Handle the service update here.
        Console.WriteLine( "UpdateType{0}\t(key={1})",
            e.UpdateType, e.Services[ 0 ].Key );
    }
}

```

Fig. 18

Docket No: MS1-1802US
 Inventor(s): Parham, et al.
 Title: SERVICE DISCOVERY AND PUBLICATION

Public NotInheritable Class SubscriptionSample

' This delegate will be called each time a notification of an
 ' update on the SampleService occurs.

Private Shared finder As New ServiceFinder

Public Shared Sub Subscribe()

' Initialize the finder to look for castle home networks.
 finder.Scopes.Add(New ConcreteScope(ConcreteScope.SsdProtocol))
 finder.Filter = New SimpleFilter(CommonServiceTypes.HomeNetwork)

' Add our handler to the delegate for service updates.
 AddHandler finder.ServiceUpdate, AddressOf Handler
 End Sub 'Subscribe

Public Shared Sub Unsubscribe()

' Remove our handler from the delegate for service updates.
 RemoveHandler finder.ServiceUpdate, AddressOf Handler
 End Sub 'Unsubscribe

' This is the callback function that was registered with the delegate.
 Private Shared Sub Handler(ByVal sender As Object, _
 ByVal e As ServiceUpdateEventArgs)

' Handle the service update here.
 Console.WriteLine("UpdateType{0}" & vbTab & "(key={1})" , _
 e.UpdateType, e.Services(0).Key)
 End Sub 'Handler

End Class 'SubscriptionSample

Fig. 19